

# DEFINIENDO LA ESTRUCTURA DE LA BASE DE DATOS

Prof. Mg. Ing. Roxana Martínez  
Prof. Ing. Damián Berrutti



**UAIOnline**  
**Ultra**»»



# DEFINIENDO LA ESTRUCTURA DE UNA BASE DE DATOS

- **OBJETIVOS**

- Comprender el Lenguaje de Definición de Datos (DDL).
- Comprender el Lenguaje de Manipulación de datos (DML).



# Definiendo la Estructura de una Base de Datos



**UAIOnline**  
**Ultra**»»



# LENGUAJE SQL

- DCL: Lenguaje de Control de Datos
- DDL: Lenguaje de Definición de Datos.
- DML: Lenguaje de Manipulación de Datos

# DCL

❖ **Lenguaje de control de datos** (*Data Control Language*) **DCL**.

encargado de la seguridad de la base de datos, en todo lo referente al control de accesos y privilegios entre los usuarios.

Como ejemplo estan : GRANT, REVOKE.

# DDL

Las sentencias SQL se clasifican según su **finalidad** dando origen a tres Sublenguajes:

❖ **Lenguaje de definicion de datos** (*Data Definition language*) **DDL**.  
es el que se encarga de la definición de la base de datos y la modificación de la estructura de los objetos que estén en ella.  
Algunos comandos propios de este sublenguaje son *CREATE*, *ALTER*, *DROP* y *TRUNCATE*

**CREATE** Utilizado para crear nuevas tablas, campos e índices

**DROP** Empleado para eliminar tablas e índices

**ALTER** Utilizado para modificar las tablas agregando campos o cambiando la definición de los campos.



# DDL - SINTAXIS Y EJEMPLO

## CREATE

### Sintaxis:

```
Create Table nombre_tabla  
(  
  nombre_campo_1 tipo_1,  
  nombre_campo_2 tipo_2,  
  nombre_campo_n tipo_n,  
  Key(campo_x,...)  
)
```

## ALTER

```
ALTER TABLE personas  
RENAME usuarios
```

Cambia el nombre de la tabla  
'personas' a 'usuarios'

Create Table pedidos

```
(  
  id_pedido INT(4) NOT NULL  
  AUTO_INCREMENT,  
  id_cliente INT(4) NOT NULL,  
  id_articulo INT(4) NOT NULL,  
  fecha DATE,  
  cantidad INT(4),  
  total INT(4),  
  KEY(id_pedido,id_cliente,id_articulo)  
)
```

Create Table articulos

```
(  
  id_articulo INT(4) NOT NULL AUTO_INCREMENT,  
  titulo VARCHAR(50),  
  autor VARCHAR(25),  
  editorial VARCHAR(25),  
  precio REAL,  
  KEY(id_articulo)  
)
```

# DDL - RESTRICCIONES (CONSTRAINTS)

## ■ Restricciones

- Se utilizan para limitar el tipo de datos en una tabla. Esto garantiza la precisión y fiabilidad de los datos en la tabla. Si hay alguna violación entre la restricción y la acción de datos, la acción se cancela.
- Las restricciones pueden ser a nivel de columna o de tabla.
  - **NOT NULL** - Asegura que una columna no permite valores Nulos.
  - **UNIQUE** - Asegura que el valor del campo es unico en toda la tabla (registros)
  - **PRIMARY KEY** - Identificación una del registro en una tabla. Es una combinación de NOT NULL y UNIQUE.
  - **FOREIGN KEY** - Identifica unicamente un registro que está en otra tabla.
  - **DEFAULT** - Se utiliza para establecer una valor por defecto en el caso que no se especifique.
  - **INDEX** - Se utiliza para realizar orden de busqueda para obtener datos rapidamente.

```
CREATE TABLE table_name (  
    column1 datatype constraint,  
    column2 datatype constraint,  
    column3 datatype constraint,  
    ....  
);
```



# DDL - RESTRICCIONES (CONSTRAINTS)

- **Restricciones**

- **NOT NULL** - Asegura que una columna no permite valores Nulos.

```
CREATE TABLE Persons (  
    ID int NOT NULL,  
    LastName varchar(255) NOT NULL,  
    FirstName varchar(255) NOT NULL,  
    Age int  
);
```

# DDL - RESTRICCIONES (CONSTRAINTS)

- **Restricciones**

- **UNIQUE** - Asegura que el valor del campo es unico en toda la tabla (registros)

```
CREATE TABLE Persons (  
    ID int NOT NULL UNIQUE,  
    LastName varchar(255) NOT NULL,  
    FirstName varchar(255),  
    Age int  
);
```

# DDL - RESTRICCIONES (CONSTRAINTS)

- **Restricciones**

- **PRIMARY KEY** - Identificación una del registro en una tabla. Es una combinación de NOT NULL y UNIQUE.

```
CREATE TABLE Persons (  
    ID int NOT NULL,  
    LastName varchar(255) NOT NULL,  
    FirstName varchar(255),  
    Age int,  
    PRIMARY KEY (ID)  
);
```

# DDL - RESTRICCIONES (CONSTRAINTS)

## ■ Restricciones

- **FOREIGN KEY** - Identifica únicamente un registro que está en otra tabla.
- Una FOREIGN KEY previene que las acciones contra los datos no destruya las relaciones entre las tablas
- También evita que se inserten datos no válidos en la columna de clave externa, porque debe ser uno de los valores contenidos en la tabla a la que apunta

PersonID	LastName	FirstName	Age
1	Hansen	Ola	30
2	Svendson	Tove	23
3	Pettersen	Kari	20

"Orders" table:

OrderID	OrderNumber	PersonID
1	77895	3
2	44678	3
3	22456	2
4	24562	1

```
CREATE TABLE Orders (  
    OrderID int NOT NULL PRIMARY KEY,  
    OrderNumber int NOT NULL,  
    PersonID int FOREIGN KEY REFERENCES Persons(PersonID)  
);
```

# DDL - RESTRICCIONES (CONSTRAINTS)

- **Restricciones**

- **DEFAULT** - Se utiliza para establecer una valor por defecto en el caso que no se especifique.

```
CREATE TABLE Persons (  
    ID int NOT NULL,  
    LastName varchar(255) NOT NULL,  
    FirstName varchar(255),  
    Age int,  
    City varchar(255) DEFAULT 'Sandnes'  
);
```

```
CREATE TABLE Orders (  
    ID int NOT NULL,  
    OrderNumber int NOT NULL,  
    OrderDate date DEFAULT GETDATE()  
);
```

# DDL - RESTRICCIONES (CONSTRAINTS)

- **Restricciones**

- **INDEX** - Se utiliza para realizar orden de búsqueda para obtener datos rápidamente.

```
CREATE INDEX index_name  
ON table_name (column1, column2, ...);
```

```
CREATE UNIQUE INDEX index_name  
ON table_name (column1, column2, ...);
```

```
CREATE INDEX idx_lastname  
ON Persons (LastName);
```

```
CREATE INDEX idx_pname  
ON Persons (LastName, FirstName);
```



# TIPO DE DATOS BÁSICOS (MSSQL SERVER)

Numérico	Alfanuméricos	Fecha	Lógico	BLOB	Otros
Integer	Char	Date	Bit	Image	Moneda
Numeric	varchar	Date Time		Text	hipervínculo
Decimal					adjunto
Float					

# DML

## ❖ *Lenguaje de Manipulación de Datos* (Data Manipulation Language) **DML**.

A través de él podemos seleccionar, insertar, eliminar y actualizar datos. Es la parte que más frecuentemente utilizaremos, y que con ella se construyen las consultas.

Algunos comandos propios de este sublenguaje son:

<b>SELECT</b>	Utilizado para consultar registros de la base de datos que satisfagan un criterio determinado
<b>INSERT</b>	Utilizado para cargar lotes de datos en la base de datos en una única operación.
<b>UPDATE</b>	Utilizado para modificar los valores de los campos y registros especificados
<b>DELETE</b>	Utilizado para eliminar registros de una tabla de una base de datos

# DML - CLÁUSULAS

Las cláusulas son condiciones de modificación utilizadas para definir los datos que desea seleccionar o manipular.

Cláusula	Descripción
FROM	Utilizada para especificar la tabla de la cual se van a seleccionar los registros
WHERE	Utilizada para especificar las condiciones que deben reunir los registros que se van a seleccionar
GROUP BY	Utilizada para separar los registros seleccionados en grupos específicos
HAVING	Utilizada para expresar la condición que debe satisfacer cada grupo
ORDER BY	Utilizada para ordenar los registros seleccionados de acuerdo con un orden específico

# DDL - CLÁUSULAS

## ■ SELECT

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
1	Alfreds Futterkiste	Maria Anders	Obere Str. 57	Berlin	12209	Germany
2	Ana Trujillo Emparedados y helados	Ana Trujillo	Avda. de la Constitución 2222	México D.F.	05021	Mexico
3	Antonio Moreno Taquería	Antonio Moreno	Mataderos 2312	México D.F.	05023	Mexico
4	Around the Horn	Thomas Hardy	120 Hanover Sq.	London	WA1 1DP	UK
5	Berglunds snabbköp	Christina Berglund	Berguvsvägen 8	Luleå	S-958 22	Sweden

```
SELECT CustomerName, City FROM Customers;
```



# DDL - CLÁUSULAS

## ■ WHERE

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
1	Alfreds Futterkiste	Maria Anders	Obere Str. 57	Berlin	12209	Germany
2	Ana Trujillo Emparedados y helados	Ana Trujillo	Avda. de la Constitución 2222	México D.F.	05021	Mexico
3	Antonio Moreno Taquería	Antonio Moreno	Mataderos 2312	México D.F.	05023	Mexico
4	Around the Horn	Thomas Hardy	120 Hanover Sq.	London	WA1 1DP	UK
5	Berglunds snabbköp	Christina Berglund	Berguvsvägen 8	Luleå	S-958 22	Sweden

```
SELECT * FROM Customers
WHERE Country='Mexico';
```

# DDL - CLÁUSULAS

- WHERE

Los predicados son condiciones que se indican en cláusula **WHERE** de una consulta SQL.

Predicados SQL	
<b>BETWEEN...AND</b>	Comprueba que al valor esta dentro de un intervalo
<b>LIKE</b>	Compara un campo con una cadena alfanumérica.
<b>ALL</b>	Señala a todos los elementos de la selección de la consulta
<b>ANY</b>	Indica que la condición se cumplirá si la comparación es cierta para al menos un elemento del conjunto.
<b>EXISTS</b>	Devuelve un valor verdadero si el resultado de una subconsulta devuelve resultados.
<b>IN</b>	Comprueba si un campo se encuentra dentro de un determinado rango. El rango puede ser una sentencia SELECT.



# DDL - CLÁUSULAS

## ■ GROUP BY

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
1	Alfreds Futterkiste	Maria Anders	Obere Str. 57	Berlin	12209	Germany
2	Ana Trujillo Emparedados y helados	Ana Trujillo	Avda. de la Constitución 2222	México D.F.	05021	Mexico
3	Antonio Moreno Taquería	Antonio Moreno	Mataderos 2312	México D.F.	05023	Mexico
4	Around the Horn	Thomas Hardy	120 Hanover Sq.	London	WA1 1DP	UK
5	Berglunds snabbköp	Christina Berglund	Berguvsvägen 8	Luleå	S-958 22	Sweden

```
SELECT column_name(s)
FROM table_name
WHERE condition
GROUP BY column_name(s)
ORDER BY column_name(s);
```

```
SELECT COUNT(CustomerID), Country
FROM Customers
GROUP BY Country;
```

# DDL - CLÁUSULAS

- **GROUP BY**

Las funciones agregadas proporcionan a SQL utilidades de cálculo sobre los datos de las tablas.

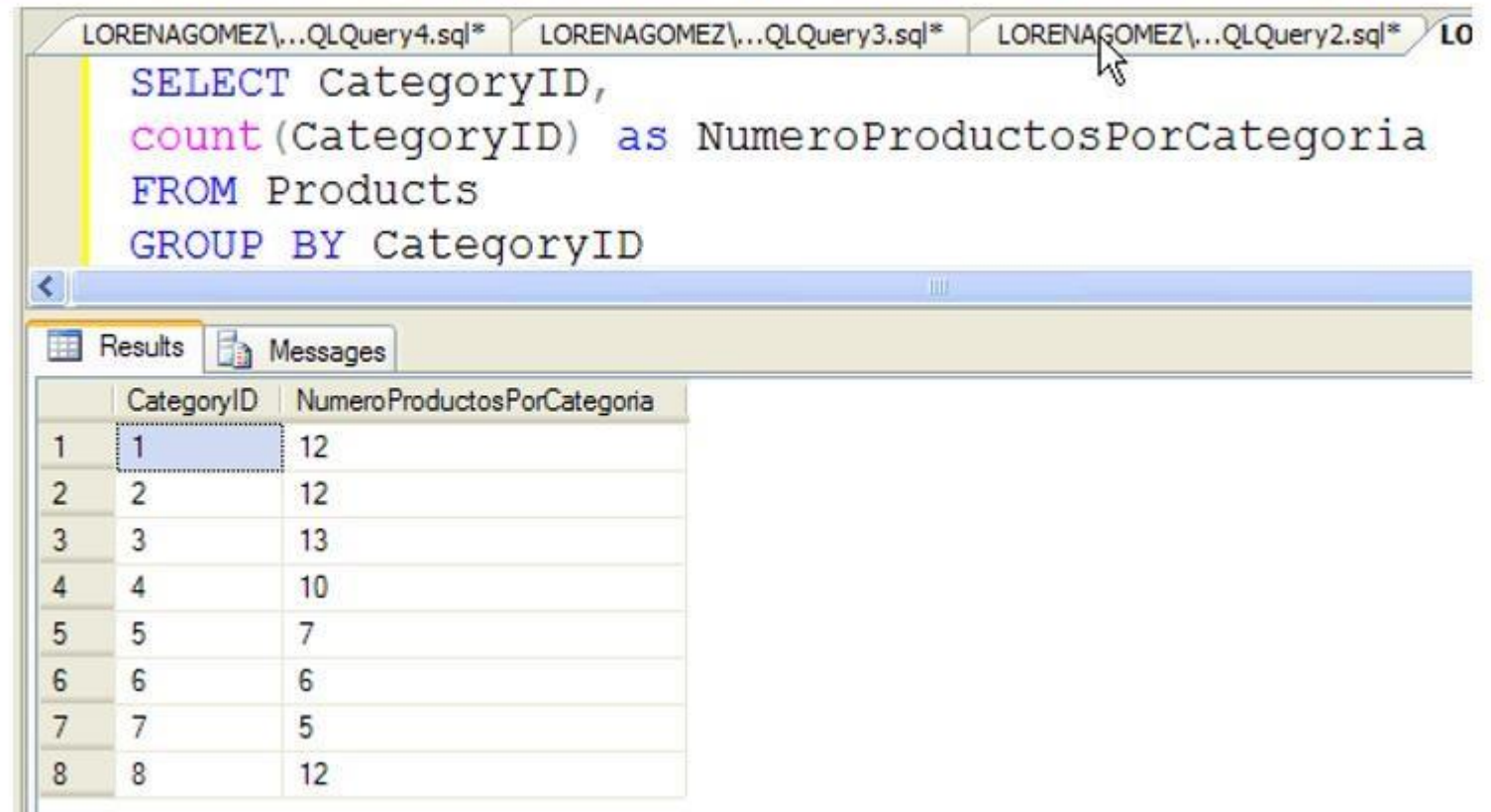
Estas funciones se incorporan en las consultas **SELECT** y retornan un **único valor** al operar sobre un grupo de registros.

Función	Descripción
AVG	Utilizada para calcular el promedio de los valores de un campo determinado
COUNT	Utilizada para devolver el número de registros de la selección
SUM	Utilizada para devolver la suma de todos los valores de un campo determinado
MAX	Utilizada para devolver el valor más alto de un campo especificado
MIN	Utilizada para devolver el valor más bajo de un campo especificado

# DDL - CLÁUSULAS

- GROUP BY
- Count
- Group By

- Se obtiene el número de productos que existen de cada categoría



The screenshot shows a SQL Server Enterprise Manager interface. At the top, there are three tabs for queries: 'LORENAGOMEZ\...QLQuery4.sql\*', 'LORENAGOMEZ\...QLQuery3.sql\*', and 'LORENAGOMEZ\...QLQuery2.sql\*'. The active query is 'LORENAGOMEZ\...QLQuery4.sql\*', which contains the following SQL code:

```
SELECT CategoryID,  
count(CategoryID) as NumeroProductosPorCategoria  
FROM Products  
GROUP BY CategoryID
```

Below the query editor, the 'Results' tab is selected, displaying the output of the query. The results are shown in a table with two columns: 'CategoryID' and 'NumeroProductosPorCategoria'. The table contains 8 rows of data.

	CategoryID	NumeroProductosPorCategoria
1	1	12
2	2	12
3	3	13
4	4	10
5	5	7
6	6	6
7	7	5
8	8	12

# DDL - CLÁUSULAS

## ■ INSERT

```
INSERT INTO table_name (column1, column2, column3, ...)
VALUES (value1, value2, value3, ...);
```

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
1	Alfreds Futterkiste	Maria Anders	Obere Str. 57	Berlin	12209	Germany
2	Ana Trujillo Emparedados y helados	Ana Trujillo	Avda. de la Constitución 2222	México D.F.	05021	Mexico
3	Antonio Moreno Taquería	Antonio Moreno	Mataderos 2312	México D.F.	05023	Mexico
4	Around the Horn	Thomas Hardy	120 Hanover Sq.	London	WA1 1DP	UK
5	Berglunds snabbköp	Christina Berglund	Berguvsvägen 8	Luleå	S-958 22	Sweden

```
INSERT INTO Customers (CustomerName, ContactName, Address, City, PostalCode, Country)
VALUES ('Cardinal', 'Tom B. Erichsen', 'Skagen 21', 'Stavanger', '4006', 'Norway');
```

```
INSERT INTO Customers (CustomerName, City, Country)
VALUES ('Cardinal', 'Stavanger', 'Norway');
```

# DDL - CLÁUSULAS

## ▪ UPDATE

```
UPDATE table_name  
SET column1 = value1, column2 = value2, ...  
WHERE condition;
```

```
UPDATE Customers  
SET ContactName = 'Alfred Schmidt', City= 'Frankfurt'  
WHERE CustomerID = 1;
```

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
1	Alfreds Futterkiste	Alfred Schmidt	Obere Str. 57	Frankfurt	12209	Germany
2	Ana Trujillo Emparedados y helados	Juan	Avda. de la Constitución 2222	México D.F.	05021	Mexico
3	Antonio Moreno Taquería	Juan	Mataderos 2312	México D.F.	05023	Mexico
4	Around the Horn	Thomas Hardy	120 Hanover Sq.	London	WA1 1DP	UK
5	Berglunds snabbköp	Christina Berglund	Berguvsvägen 8	Luleå	S-958 22	Sweden



# DDL - CLÁUSULAS

- **DELETE** `DELETE FROM table_name WHERE condition;`

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
1	Alfreds Futterkiste	Maria Anders	Obere Str. 57	Berlin	12209	Germany
2	Ana Trujillo Emparedados y helados	Ana Trujillo	Avda. de la Constitución 2222	México D.F.	05021	Mexico
3	Antonio Moreno Taquería	Antonio Moreno	Mataderos 2312	México D.F.	05023	Mexico
4	Around the Horn	Thomas Hardy	120 Hanover Sq.	London	WA1 1DP	UK
5	Berglunds snabbköp	Christina Berglund	Berguvsvägen 8	Luleå	S-958 22	Sweden

```
DELETE FROM Customers WHERE CustomerName='Alfreds Futterkiste';
```



# DDL - CLÁUSULAS

- **DELETE** | `DELETE FROM table_name;`

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
1	Alfreds Futterkiste	Maria Anders	Obere Str. 57	Berlin	12209	Germany
2	Ana Trujillo Emparedados y helados	Ana Trujillo	Avda. de la Constitución 2222	México D.F.	05021	Mexico
3	Antonio Moreno Taquería	Antonio Moreno	Mataderos 2312	México D.F.	05023	Mexico
4	Around the Horn	Thomas Hardy	120 Hanover Sq.	London	WA1 1DP	UK
5	Berglunds snabbköp	Christina Berglund	Berguvsvägen 8	Luleå	S-958 22	Sweden

`DELETE FROM Customers;`



# EJEMPLOS DE SELECCION

Hay que empezar por la palabra **SELECT**, después puedes poner **ALL(Predicado)** o nada, a continuación un nombre de columna(Campos), o varios separados por comas(,), a continuación la palabra **FROM(Clausula)** y una expresión-tabla, y por último de forma opcional puedes incluir la cláusula **WHERE** con una condición-de-búsqueda.

```
SELECT Campos FROM Tabla  
SELECT Nombre, Teléfono FROM Clientes
```

```
SELECT CodigoPostal, Nombre,  
Telefono  
FROM Clientes  
ORDER BY Nombre
```

Consulta para traer un registro cuyo campo empiece por determinado valor. Ejemplo: traer nombre del cliente donde el campo de dirección empiece por AV, se usa clausula LIKE

```
SELECT NOMBRE_CLIENTE FROM TABLA_CLIENTES WHERE  
CAMPO_DIRECCION LIKE "AV%".
```

# EJEMPLOS DE SELECCION

Consulta para traer los registros cuyo campo buscado este dentro de un rango dado.

Ejemplo: traer los registros de las facturas cuyos números de facturas estén entre 102 y 118. Para lo cual se usa la clausula BETWEEN.

```
SELECT * FROM TABLA_FACTURAS WHERE CODIGO_FACTURA  
BETWEEN 102 and 118.
```

Consulta para sumar un campo de una tabla:

```
SELECT SUM (CAMPO_VALOR) FROM TABLA_TOTALES
```

Insertar registros en una tabla:

```
INSERT INTO "nombre_tabla" ("columna1", "columna2", ...)  
VALUES ("valor1", "valor2", ...)
```

```
INSERT INTO "Estudiante" (Nombre, Apellido, ...)  
VALUES (Andres, wood,...)
```

# EJEMPLOS DE SELECCIÓN (DISTINCT)

Mostrar en que ciudades  
hay clientes

LORENAGOMEZ\...QLQuery1.sql\*

```
select city  
from Customers
```

Results Messages

	City
1	Aachen
2	Albuquerque
3	Anchorage
4	Arhus
5	Barcelona
6	Barquisimeto
7	Bergamo
8	Berlin
9	Bern
10	Boise
11	Bräcke
12	Brandenburg
13	Bruxelles
14	Buenos Aires
15	Buenos Aires
16	Buenos Aires
17	Butte
18	Campinas

Se repiten las ciudades

Usar **DISTINCT**

No se repiten

LORENAGOMEZ\...QLQuery1.sql\* Summary

```
select distinct city  
from Customers
```

Results Messages

	City
1	Aachen
2	Albuquerque
3	Anchorage
4	Arhus
5	Barcelona
6	Barquisimeto
7	Bergamo
8	Berlin
9	Bern
10	Boise
11	Bräcke
12	Brandenburg
13	Bruxelles
14	Buenos Aires
15	Butte
16	Campinas
17	Caracas
18	Charleroi
19	Cork

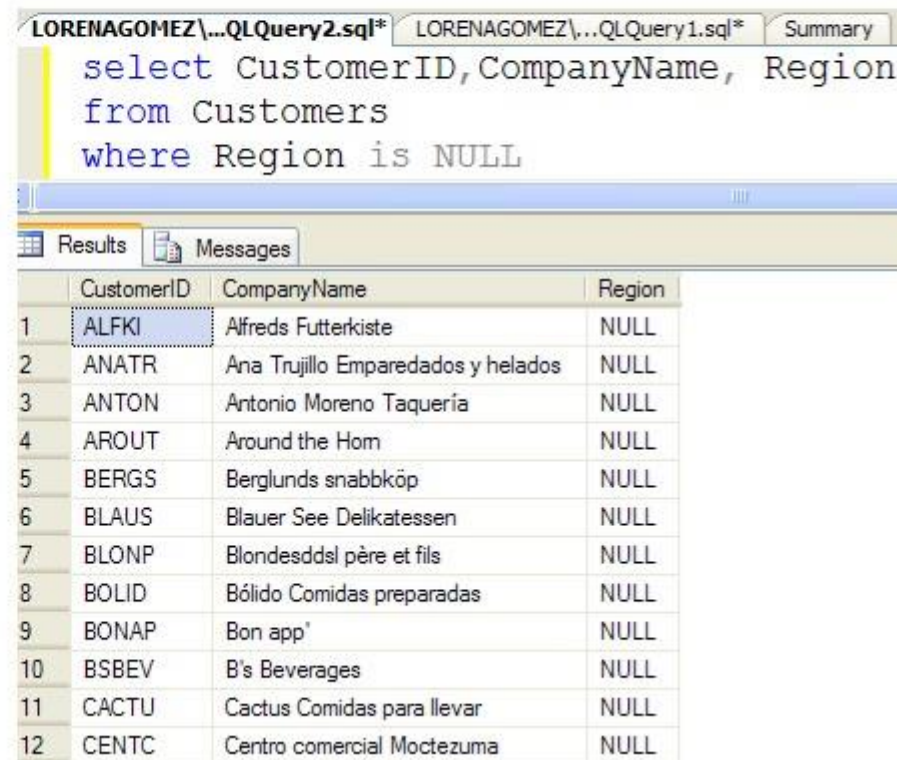


# EJEMPLOS DE SELECCIÓN (NULL)

- Obtener Id del cliente, nombre de la compañía y la Region para aquellos clientes cuya Region sea NULL

Null

Valor que significa que al atributo no se le asignó un valor o se le asignó NULL



The screenshot shows a SQL query window with the following text:

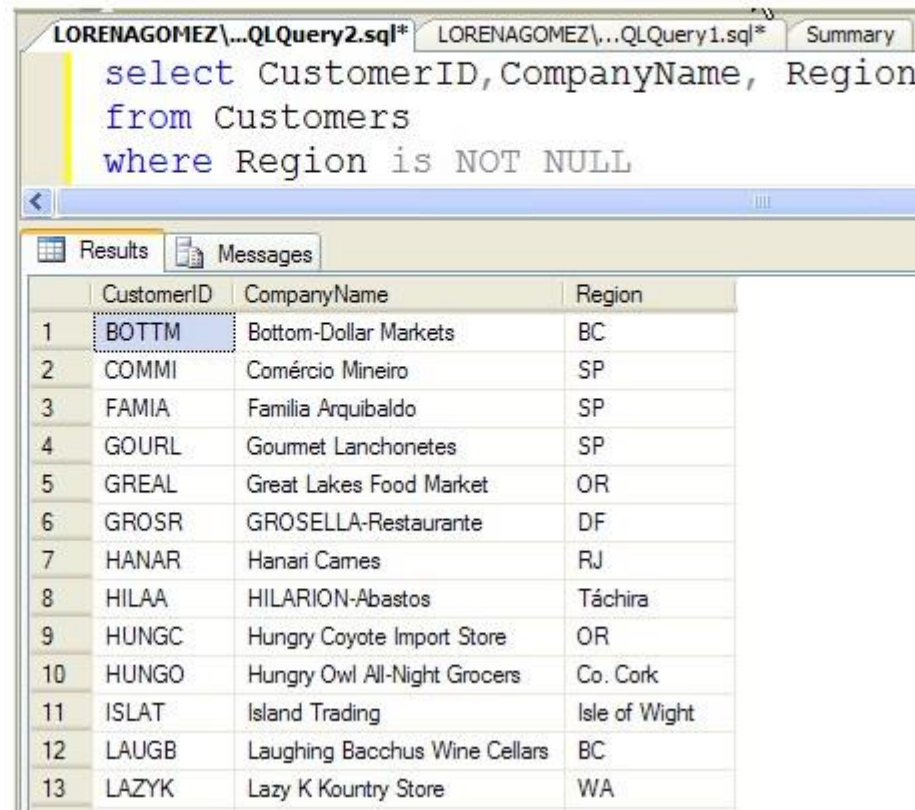
```
select CustomerID, CompanyName, Region  
from Customers  
where Region is NULL
```

Below the query, the 'Results' tab is active, displaying a table with 12 rows. The first row is highlighted. The table columns are CustomerID, CompanyName, and Region.

	CustomerID	CompanyName	Region
1	ALFKI	Alfreds Futterkiste	NULL
2	ANATR	Ana Trujillo Emparedados y helados	NULL
3	ANTON	Antonio Moreno Taquería	NULL
4	AROUT	Around the Horn	NULL
5	BERGS	Berglunds snabbköp	NULL
6	BLAUS	Blauer See Delikatessen	NULL
7	BLONP	Blondesddsl père et fils	NULL
8	BOLID	Bólido Comidas preparadas	NULL
9	BONAP	Bon app'	NULL
10	BSBEV	B's Beverages	NULL
11	CACTU	Cactus Comidas para llevar	NULL
12	CENTC	Centro comercial Moctezuma	NULL

# EJEMPLOS DE SELECCIÓN (NOT NULL)

- Obtener Id del cliente, nombre de la compañía y la Region para aquellos clientes cuya Region tenga un valor asignado



The screenshot shows a SQL Server Enterprise Manager window with a query editor and a results pane. The query editor displays the following SQL statement:

```
select CustomerID, CompanyName, Region
from Customers
where Region is NOT NULL
```

The results pane shows a table with 13 rows of data. The first row is highlighted. The columns are CustomerID, CompanyName, and Region.

	CustomerID	CompanyName	Region
1	BOTTM	Bottom-Dollar Markets	BC
2	COMMI	Comércio Mineiro	SP
3	FAMIA	Familia Arquibaldo	SP
4	GOURL	Gourmet Lanchonetes	SP
5	GREAL	Great Lakes Food Market	OR
6	GROSR	GROSELLA-Restaurante	DF
7	HANAR	Hanari Cames	RJ
8	HILAA	HILARION-Abastos	Táchira
9	HUNGC	Hungry Coyote Import Store	OR
10	HUNGO	Hungry Owl All-Night Grocers	Co. Cork
11	ISLAT	Island Trading	Isle of Wight
12	LAUGB	Laughing Bacchus Wine Cellars	BC
13	LAZYK	Lazy K Kountry Store	WA



# OPERADORES LÓGICOS

Los operadores lógicos permiten comparar expresiones lógicas devolviendo siempre un valor verdadero o falso. Los operadores lógicos se evalúan de izquierda a derecha.

Operador	Uso
AND	Es el "y" lógico. Evalúa dos condiciones y devuelve un valor de verdad sólo si ambas son ciertas.
OR	Es el "o" lógico. Evalúa dos condiciones y devuelve un valor de verdad si alguna de las dos es cierta.
NOT	Negación lógica. Devuelve el valor contrario de la expresión.
+ (Concatenación)	Se usa para unir datos de tipo alfanumérico

# OPERADORES LÓGICOS

## AND Syntax

```
SELECT column1, column2, ...  
FROM table_name  
WHERE condition1 AND condition2 AND condition3 ...;
```

## OR Syntax

```
SELECT column1, column2, ...  
FROM table_name  
WHERE condition1 OR condition2 OR condition3 ...;
```

## NOT Syntax

```
SELECT column1, column2, ...  
FROM table_name  
WHERE NOT condition;
```

# OPERADORES LÓGICOS

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
1	Alfreds Futterkiste	Maria Anders	Obere Str. 57	Berlin	12209	Germany
2	Ana Trujillo Emparedados y helados	Ana Trujillo	Avda. de la Constitución 2222	México D.F.	05021	Mexico
3	Antonio Moreno Taquería	Antonio Moreno	Mataderos 2312	México D.F.	05023	Mexico
4	Around the Horn	Thomas Hardy	120 Hanover Sq.	London	WA1 1DP	UK
5	Berglunds snabbköp	Christina Berglund	Berguvsvägen 8	Luleå	S-958 22	Sweden
6	Blauer See Delikatessen	Hanna Moos	Forsterstr. 57	Mannheim	68306	Germany
7	Blondel père et fils	Frédérique Citeaux	24, place Kléber	Strasbourg	67000	France
8	Bólido Comidas preparadas	Martín Sommer	C/ Araquil, 67	Madrid	28023	Spain
9	Bon app'	Laurence Lebihans	12, rue des Bouchers	Marseille	13008	France
10	Bottom-Dollar Marketse	Elizabeth Lincoln	23 Tsawassen Blvd.	Tsawassen	T2F 8M4	Canada
11	B's Beverages	Victoria Ashworth	Fauntleroy Circus	London	EC2 5NT	UK

```
SELECT * FROM Customers
WHERE Country='Germany' AND City='Berlin';
```

```
SELECT * FROM Customers
WHERE Country='Germany' OR Country='Spain';
```

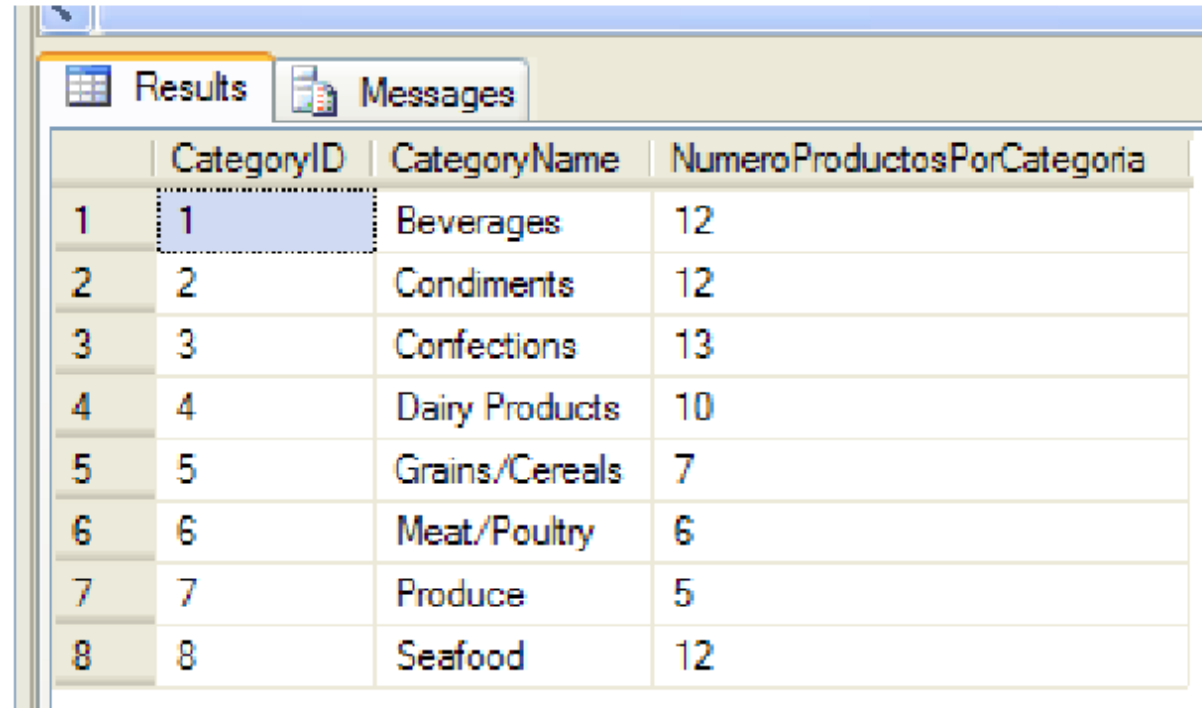
```
SELECT * FROM Customers
WHERE NOT Country='Germany';
```

# OPERADORES RELACIONALES

Operador	Uso
<	Menor que
>	Mayor que
<>    !=	Distinto de
<=	Menor o igual que
>=	Mayor o igual que
=	Igual que

# JOINS

- El tener un número de categoría no indica nada, es mejor poner el nombre de la categoría como en el resultado siguiente:



	CategoryID	CategoryName	NumeroProductosPorCategoria
1	1	Beverages	12
2	2	Condiments	12
3	3	Confections	13
4	4	Dairy Products	10
5	5	Grains/Cereals	7
6	6	Meat/Poultry	6
7	7	Produce	5
8	8	Seafood	12

- Pero, como se escribe el query? Con un JOIN entre la llave foránea FK IDCATEGORY y la llave Primaria Categories(IDCATEGORY)



# JOINS

- El nombre de la categoría está en la tabla **Categories** y nuestra consulta utiliza la tabla de **Productos**
- Poner en el query las 2 tablas y especificar una condición donde la **FK=PK**

Table - dbo.Categories			Table - dbo.Products	Diagram - L...
CategoryID	CategoryName	Description		
1	Beverages	Soft drinks, coff...		
2	Condiments	Sweet and savo...		
3	Confections	Desserts, candie...		
4	Dairy Products	Cheeses		
5	Grains/Cereals	Breads, crackers...		
6	Meat/Poultry	Prepared meats		
7	Produce	Dried fruit and b...		
8	Seafood	Seaweed and fish		

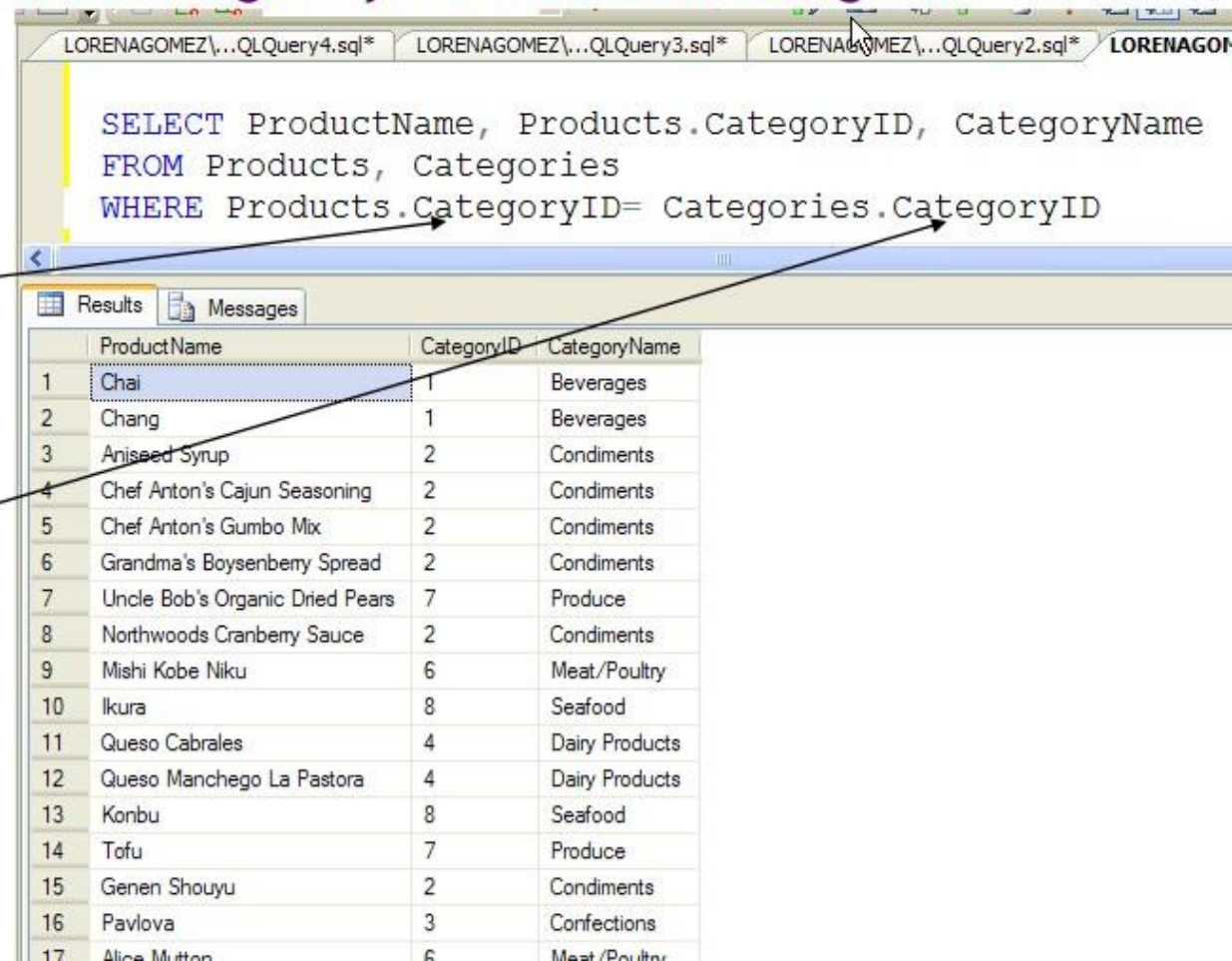
Table - dbo.Products		Diagram - LOR...ind.Diagram_0		Summary					
	ProductID	ProductName	SupplierID	CategoryID	QuantityPerUnit	UnitPrice	UnitsInStock	UnitsOnOrder	ReorderLevel
▶	1	Chai	1	1	10 boxes x 20 b...	18.0000	39	0	10
	2	Chang	1	1	24 - 12 oz bottles	19.0000	17	40	25
	3	Aniseed Syrup	1	2	12 - 550 ml bottles	10.0000	13	70	25
	4	Chef Anton's Cajun Seasoning	2	2	48 - 6 oz jars	22.0000	53	0	0
	5	Chef Anton's Gumbo Mix	2	2	36 boxes	21.3500	0	0	0
	6	Grandma's Boysenberry Spread	3	2	12 - 8 oz jars	25.0000	120	0	25
	7	Uncle Bob's Organic Dried Pears	3	7	12 - 1 lb pkgs.	30.0000	15	0	10
	8	Northwoods Cranberry Sauce	3	2	12 - 12 oz jars	40.0000	6	0	0
	9	Mishi Kobe Niku	4	6	18 - 500 g pkgs.	97.0000	29	0	0
	10	Ikura	4	8	12 - 200 ml jars	31.0000	31	0	0

# JOINS

Nombre del producto, Id de categoría y Nombre de la categoría

FK

PK

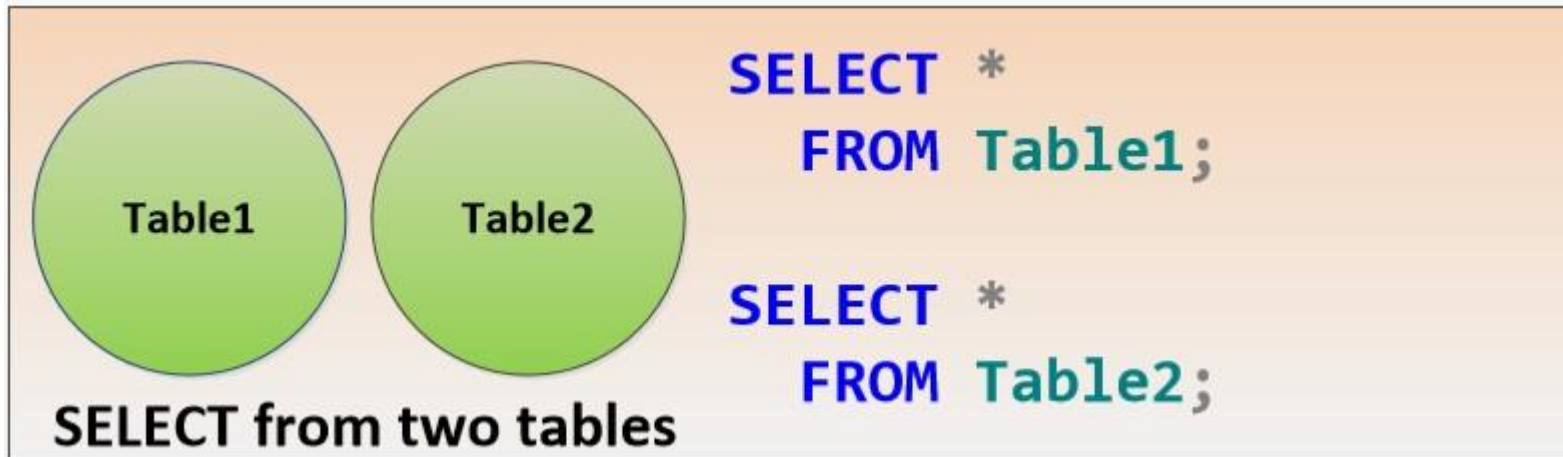


```
SELECT ProductName, Products.CategoryID, CategoryName
FROM Products, Categories
WHERE Products.CategoryID= Categories.CategoryID
```

	ProductName	CategoryID	CategoryName
1	Chai	1	Beverages
2	Chang	1	Beverages
3	Aniseed Syrup	2	Condiments
4	Chef Anton's Cajun Seasoning	2	Condiments
5	Chef Anton's Gumbo Mix	2	Condiments
6	Grandma's Boysenberry Spread	2	Condiments
7	Uncle Bob's Organic Dried Pears	7	Produce
8	Northwoods Cranberry Sauce	2	Condiments
9	Mishi Kobe Niku	6	Meat/Poultry
10	Ikura	8	Seafood
11	Queso Cabrales	4	Dairy Products
12	Queso Manchego La Pastora	4	Dairy Products
13	Konbu	8	Seafood
14	Tofu	7	Produce
15	Genen Shouyu	2	Condiments
16	Pavlova	3	Confections
17	Alice Mutton	6	Meat/Poultry

# JOINS

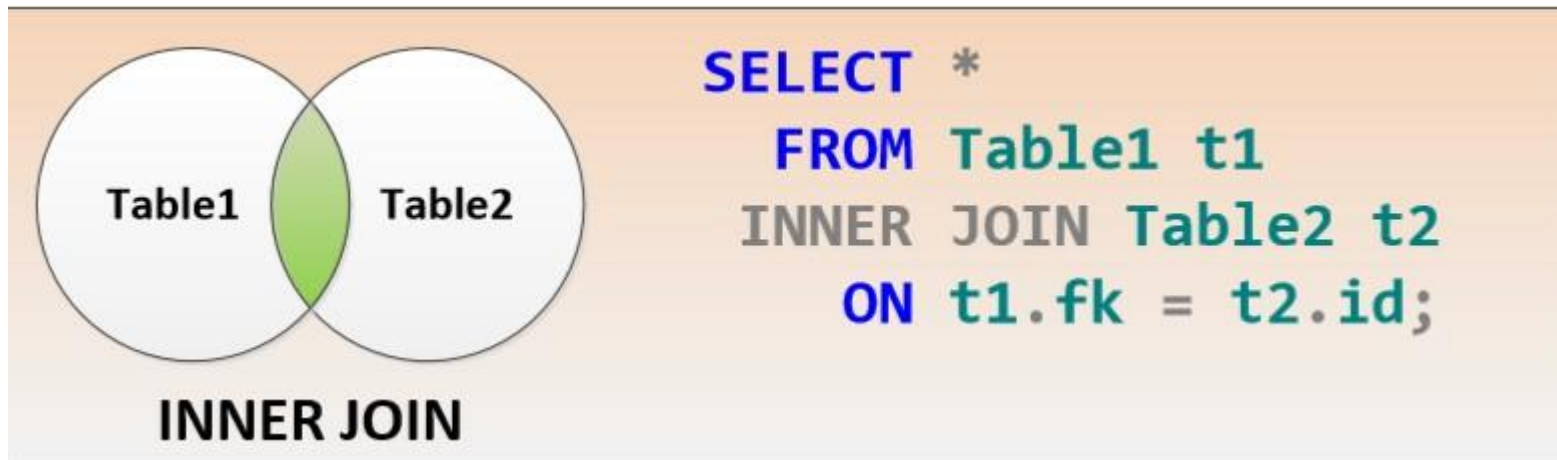
La cláusula JOIN se usa para combinar filas de dos o más tablas, en función de una columna relacionada entre ellas.





# JOINS - INNER JOIN

INNER JOIN selecciona registros que tienen valores coincidentes en ambas tablas.



# JOINS - INNER JOIN

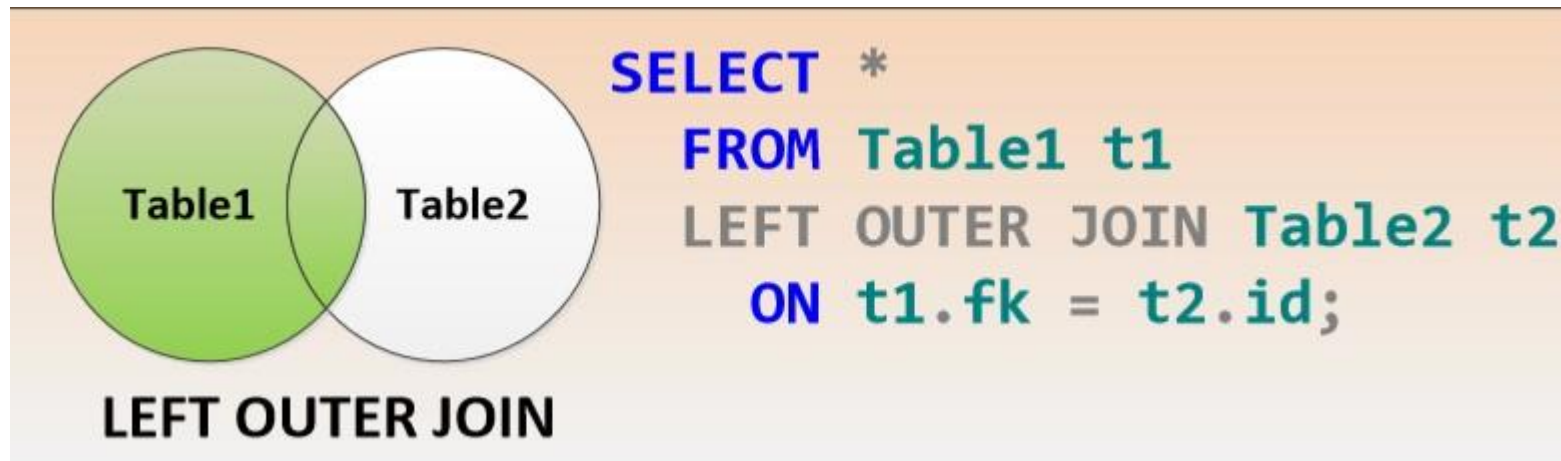
OrderID	CustomerID	EmployeeID	OrderDate	ShipperID
10308	2	7	1996-09-18	3
10309	37	3	1996-09-19	1
10310	77	8	1996-09-20	2

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
1	Alfreds Futterkiste	Maria Anders	Obere Str. 57	Berlin	12209	Germany
2	Ana Trujillo Emparedados y helados	Ana Trujillo	Avda. de la Constitución 2222	México D.F.	05021	Mexico
3	Antonio Moreno Taquería	Antonio Moreno	Mataderos 2312	México D.F.	05023	Mexico

```
SELECT Orders.OrderID, Customers.CustomerName
FROM Orders
INNER JOIN Customers ON Orders.CustomerID = Customers.CustomerID;
```

# JOINS - LEFT JOIN

LEFT JOIN devuelve todos los registros de la tabla izquierda (tabla1) y los registros coincidentes de la tabla derecha (tabla2). El resultado es NULL desde el lado derecho, si no hay coincidencia.



# JOINS - LEFT OUTER JOIN

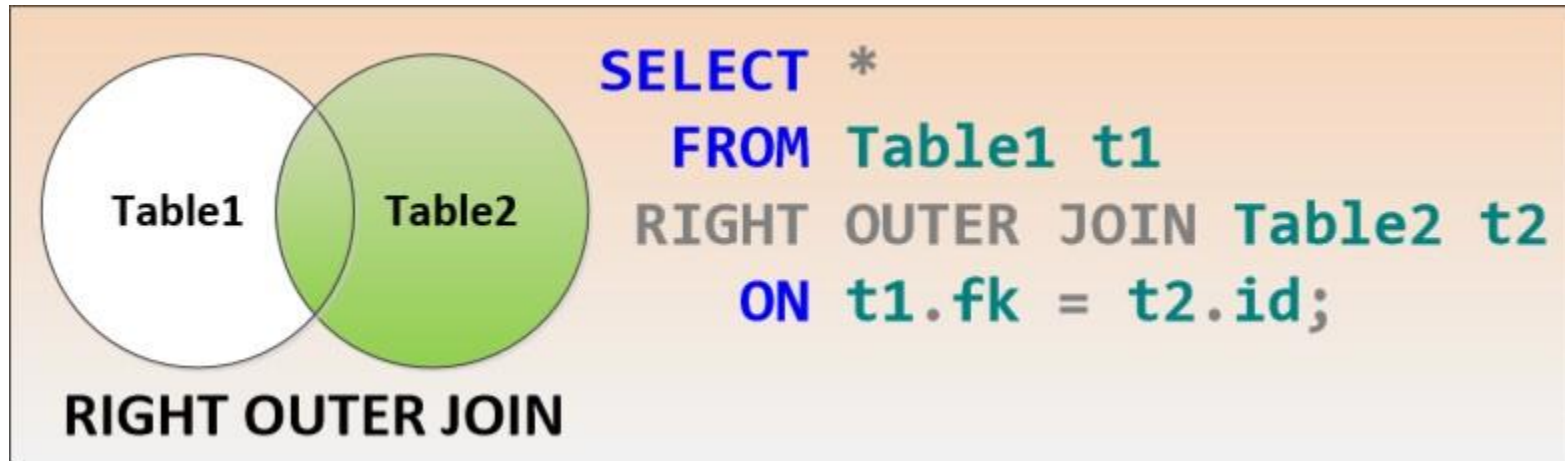
CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
1	Alfreds Futterkiste	Maria Anders	Obere Str. 57	Berlin	12209	Germany
2	Ana Trujillo Emparedados y helados	Ana Trujillo	Avda. de la Constitución 2222	México D.F.	05021	Mexico
3	Antonio Moreno Taquería	Antonio Moreno	Mataderos 2312	México D.F.	05023	Mexico

OrderID	CustomerID	EmployeeID	OrderDate	ShipperID
10308	2	7	1996-09-18	3
10309	37	3	1996-09-19	1
10310	77	8	1996-09-20	2

```
SELECT Customers.CustomerName, Orders.OrderID
FROM Customers
LEFT JOIN Orders ON Customers.CustomerID = Orders.CustomerID
ORDER BY Customers.CustomerName;
```

# JOINS - RIGHT OUTER JOIN

RIGHT JOIN devuelve todos los registros de la tabla derecha (tabla2) y los registros coincidentes de la tabla izquierda (tabla1). El resultado es NULL desde el lado izquierdo, cuando no hay coincidencia.



# JOINS - RIGHT OUTER JOIN

OrderID	CustomerID	EmployeeID	OrderDate	ShipperID
10308	2	7	1996-09-18	3
10309	37	3	1996-09-19	1
10310	77	8	1996-09-20	2

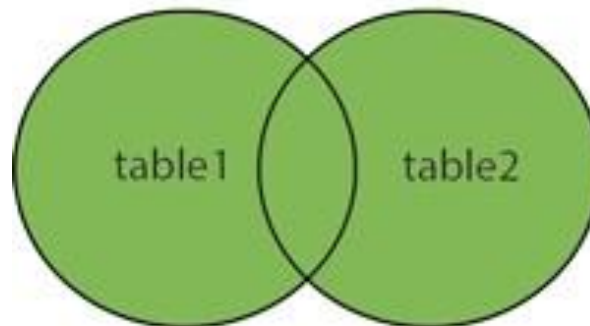
EmployeeID	LastName	FirstName	BirthDate	Photo
1	Davolio	Nancy	12/8/1968	EmpID1.pic
2	Fuller	Andrew	2/19/1952	EmpID2.pic
3	Leverling	Janet	8/30/1963	EmpID3.pic

```
SELECT Orders.OrderID, Employees.LastName, Employees.FirstName
FROM Orders
RIGHT JOIN Employees ON Orders.EmployeeID = Employees.EmployeeID
ORDER BY Orders.OrderID;
```



# JOINS - FULL OUTER JOIN

FULL OUTER JOIN



```
SELECT column_name(s)
FROM table1
FULL OUTER JOIN table2
ON table1.column_name = table2.column_name
WHERE condition;
```

# JOINS - FULL OUTER JOIN

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
1	Alfreds Futterkiste	Maria Anders	Obere Str. 57	Berlin	12209	Germany
2	Ana Trujillo Emparedados y helados	Ana Trujillo	Avda. de la Constitución 2222	México D.F.	05021	Mexico
3	Antonio Moreno Taquería	Antonio Moreno	Mataderos 2312	México D.F.	05023	Mexico

OrderID	CustomerID	EmployeeID	OrderDate	ShipperID
10308	2	7	1996-09-18	3
10309	37	3	1996-09-19	1
10310	77	8	1996-09-20	2

```
SELECT Customers.CustomerName, Orders.OrderID
FROM Customers
FULL OUTER JOIN Orders ON Customers.CustomerID=Orders.CustomerID
ORDER BY Customers.CustomerName;
```

CustomerName	OrderID
Alfreds Futterkiste	<i>Null</i>
Ana Trujillo Emparedados y helados	10308
Antonio Moreno Taquería	10365

# SUBCONSULTAS

- **OPERADORES EN SUBCONSULTAS**

- Se pueden usar operadores como :
  - **IN**
  - **EXISTS**
  - **ANY / ALL**
  - **SELECT INTO**
  - **SELECT INSERT INTO**

# SUBCONSULTAS

## ▪ OPERADORES EN SUBCONSULTAS

### ▪ IN

- El operador IN le permite especificar múltiples valores en una cláusula WHERE.
- El operador IN es una abreviatura para múltiples condiciones OR
- Se pueden usar como Subconsulta

```
SELECT column_name(s)  
FROM table_name  
WHERE column_name IN (value1, value2, ...);
```

```
SELECT column_name(s)  
FROM table_name  
WHERE column_name IN (SELECT STATEMENT);
```

# SUBCONSULTAS

- OPERADORES EN SUBCONSULTAS

- IN

SupplierID	SupplierName	ContactName	Address	City	PostalCode	Country
1	Exotic Liquid	Charlotte Cooper	49 Gilbert St.	London	EC1 4SD	UK
2	New Orleans Cajun Delights	Shelley Burke	P.O. Box 78934	New Orleans	70117	USA
3	Grandma Kelly's Homestead	Regina Murphy	707 Oxford Rd.	Ann Arbor	48104	USA

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
1	Alfreds Futterkiste	Maria Anders	Obere Str. 57	Berlin	12209	Germany
2	Ana Trujillo Emparedados y helados	Ana Trujillo	Avda. de la Constitución 2222	México D.F.	05021	Mexico
3	Antonio Moreno Taquería	Antonio Moreno	Mataderos 2312	México D.F.	05023	Mexico
4	Around the Horn	Thomas Hardy	120 Hanover Sq.	London	WA1 1DP	UK
5	Berglunds snabbköp	Christina Berglund	Berguvsvägen 8	Luleå	S-958 22	Sweden
6	Blauer See Delikatessen	Hanna Moos	Forsterstr. 57	Mannheim	68306	Germany
7	Blondel père et fils	Frédérique Citeaux	24, place Kléber	Strasbourg	67000	France
8	Bóldo Comidas preparadas	Martín Sommer	C/ Araquil, 67	Madrid	28023	Spain
9	Bon app'	Laurence Lebihans	12, rue des Bouchers	Marseille	13008	France
10	Bottom-Dollar Marketse	Elizabeth Lincoln	23 Tsawassen Blvd.	Tsawassen	T2F 8M4	Canada
11	B's Beverages	Victoria Ashworth	Fauntleroy Circus	London	EC2 5NT	UK

```
SELECT * FROM Customers
WHERE Country IN (SELECT Country FROM Suppliers);
```

# SUBCONSULTAS

- **OPERADORES EN SUBCONSULTAS**

- **EXISTS**

- El Operador EXISTS se usa para comprobar la existencia de cualquier registro en una subconsulta.
    - Devuelve verdadero si la subconsulta devuelve uno o más registros.

```
SELECT column_name(s)
FROM table_name
WHERE EXISTS
(SELECT column_name FROM table_name WHERE condition);
```



# SUBCONSULTAS

## ■ OPERADORES EN SUBCONSULTAS

### ■ EXISTS

ProductID	ProductName	SupplierID	CategoryID	Unit	Price
1	Chais	1	1	10 boxes x 20 bags	18
2	Chang	1	1	24 - 12 oz bottles	19
3	Aniseed Syrup	1	2	12 - 550 ml bottles	10
4	Chef Anton's Cajun Seasoning	2	2	48 - 6 oz jars	22
5	Chef Anton's Gumbo Mix	2	2	36 boxes	21.35

SupplierID	SupplierName	ContactName	Address	City	PostalCode	Country
1	Exotic Liquid	Charlotte Cooper	49 Gilbert St.	London	EC1 4SD	UK
2	New Orleans Cajun Delights	Shelley Burke	P.O. Box 78934	New Orleans	70117	USA
3	Grandma Kelly's Homestead	Regina Murphy	707 Oxford Rd.	Ann Arbor	48104	USA
4	Tokyo Traders	Yoshi Nagase	9-8 Sekimai Musashino-shi	Tokyo	100	Japan

```
SELECT SupplierName
FROM Suppliers
WHERE EXISTS (SELECT ProductName FROM Products WHERE Products.SupplierID = Suppliers.supplierID AND Price < 20);
```

```
SELECT SupplierName
FROM Suppliers
WHERE EXISTS (SELECT ProductName FROM Products WHERE Products.SupplierID = Suppliers.supplierID AND Price = 22);
```

# SUBCONSULTAS

## ■ OPERADORES EN SUBCONSULTAS

### ■ ANY / ALL

- Los operadores ANY y ALL se usan con una cláusula WHERE o HAVING.
- El operador ANY devuelve verdadero si alguno de los valores de subconsulta cumple la condición.
- El operador ALL devuelve verdadero si todos los valores de subconsulta cumplen la condición.
- Sintaxis:

```
SELECT column_name(s)
FROM table_name
WHERE column_name operator ANY
(SELECT column_name FROM table_name WHERE condition);
```

```
SELECT column_name(s)
FROM table_name
WHERE column_name operator ALL
(SELECT column_name FROM table_name WHERE condition);
```

# SUBCONSULTAS

## ■ OPERADORES EN SUBCONSULTAS

### ■ ANY / ALL

ProductID	ProductName	SupplierID	CategoryID	Unit	Price
1	Chais	1	1	10 boxes x 20 bags	18
2	Chang	1	1	24 - 12 oz bottles	19
3	Aniseed Syrup	1	2	12 - 550 ml bottles	10
4	Chef Anton's Cajun Seasoning	2	2	48 - 6 oz jars	22
5	Chef Anton's Gumbo Mix	2	2	36 boxes	21.35

OrderDetailID	OrderID	ProductID	Quantity
1	10248	11	12
2	10248	42	10
3	10248	72	5
4	10249	14	9
5	10249	51	40

```
SELECT ProductName
FROM Products
WHERE ProductID = ANY (SELECT ProductID FROM OrderDetails WHERE Quantity = 10);
```

Devuelve TRUE y enumera los nombres de productos si encuentra CUALQUIER registro en la tabla OrderDetails esa cantidad = 10



```
SELECT ProductName
FROM Products
WHERE ProductID = ALL (SELECT ProductID FROM OrderDetails WHERE Quantity = 10);
```



Devuelve VERDADERO y enumera los nombres de productos si TODOS los registros en la tabla OrderDetails tienen cantidad = 10 (por lo tanto, este ejemplo devolverá FALSE, porque no TODOS los registros en la tabla OrderDetails tienen cantidad = 10):

# SUBCONSULTAS

## ■ OPERADORES EN SUBCONSULTAS

### ■ SELECT INTO

- La instrucción SELECT INTO copia datos de una tabla en una nueva tabla.

```
SELECT *  
INTO newtable [IN externaldb]  
FROM oldtable  
WHERE condition;
```

Tabla Entera

```
SELECT column1, column2, column3, ...  
INTO newtable [IN externaldb]  
FROM oldtable  
WHERE condition;
```

Multiples  
Campos

Multiples  
Campos - Una  
Tabla

```
SELECT * INTO CustomersBackup2017  
FROM Customers;
```

Multiples  
Campos y  
Multiples Tablas

```
SELECT CustomerName, ContactName INTO CustomersBackup2017  
FROM Customers;
```

```
SELECT Customers.CustomerName, Orders.OrderID  
INTO CustomersOrderBackup2017  
FROM Customers  
LEFT JOIN Orders ON Customers.CustomerID = Orders.CustomerID;
```

# SUBCONSULTAS

## ■ OPERADORES EN SUBCONSULTAS

### ■ SELECT INTO

- La instrucción SELECT INTO copia datos de una tabla en una nueva tabla.

```
SELECT *  
INTO newtable [IN externaldb]  
FROM oldtable  
WHERE condition;
```

Tabla Entera

```
SELECT column1, column2, column3, ...  
INTO newtable [IN externaldb]  
FROM oldtable  
WHERE condition;
```

Multiples  
Campos

Multiples  
Campos - Una  
Tabla

```
SELECT * INTO CustomersBackup2017  
FROM Customers;
```

Multiples  
Campos y  
Multiples Tablas

```
SELECT CustomerName, ContactName INTO CustomersBackup2017  
FROM Customers;
```

```
SELECT Customers.CustomerName, Orders.OrderID  
INTO CustomersOrderBackup2017  
FROM Customers  
LEFT JOIN Orders ON Customers.CustomerID = Orders.CustomerID;
```



# SUBCONSULTAS

## ■ OPERADORES EN SUBCONSULTAS

### ■ SELECT INTO

- La instrucción SELECT INTO copia datos de una tabla en una nueva tabla.

```
SELECT *  
INTO newtable [IN externaldb]  
FROM oldtable  
WHERE condition;
```

Tabla Entera

```
SELECT column1, column2, column3, ...  
INTO newtable [IN externaldb]  
FROM oldtable  
WHERE condition;
```

Multiples  
Campos

Multiples  
Campos - Una  
Tabla

```
SELECT * INTO CustomersBackup2017  
FROM Customers;
```

Multiples  
Campos y  
Multiples Tablas

```
SELECT CustomerName, ContactName INTO CustomersBackup2017  
FROM Customers;
```

```
SELECT Customers.CustomerName, Orders.OrderID  
INTO CustomersOrderBackup2017  
FROM Customers  
LEFT JOIN Orders ON Customers.CustomerID = Orders.CustomerID;
```

# SUBCONSULTAS

- **OPERADORES EN SUBCONSULTAS**

- **INSERT INTO SELECT**

- La instrucción INSERT INTO SELECT copia datos de una tabla y los inserta en otra tabla.
    - INSERT INTO SELECT requiere que los tipos de datos en las tablas de origen y destino coincidan
    - Los registros existentes en la tabla de origen no se ven afectados

```
INSERT INTO table2
SELECT * FROM table1
WHERE condition;
```

```
INSERT INTO table2 (column1, column2, column3, ...)
SELECT column1, column2, column3, ...
FROM table1
WHERE condition;
```

# SUBCONSULTAS

- OPERADORES EN SUBCONSULTAS

- INSERT INTO SELECT

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
1	Alfreds Futterkiste	Maria Anders	Obere Str. 57	Berlin	12209	Germany
2	Ana Trujillo Emparedados y helados	Ana Trujillo	Avda. de la Constitución 2222	México D.F.	05021	Mexico
3	Antonio Moreno Taquería	Antonio Moreno	Mataderos 2312	México D.F.	05023	Mexico

SupplierID	SupplierName	ContactName	Address	City	Postal Code	Country
1	Exotic Liquid	Charlotte Cooper	49 Gilbert St.	Londona	EC1 4SD	UK
2	New Orleans Cajun Delights	Shelley Burke	P.O. Box 78934	New Orleans	70117	USA
3	Grandma Kelly's Homestead	Regina Murphy	707 Oxford Rd.	Ann Arbor	48104	USA

```
INSERT INTO Customers (CustomerName, City, Country)
SELECT SupplierName, City, Country FROM Suppliers;
```

```
INSERT INTO Customers (CustomerName, ContactName, Address, City, PostalCode, Country)
SELECT SupplierName, ContactName, Address, City, PostalCode, Country FROM Suppliers;
```

```
INSERT INTO Customers (CustomerName, City, Country)
SELECT SupplierName, City, Country FROM Suppliers
WHERE Country='Germany';
```



¿Consultas?



**UAI**

**Universidad Abierta  
Interamericana**